(12) UK Patent Application (19) GB (11) 2 336 005 (13) A

(43) Date of A Publication 06.10.1999

(21) Application No 9806575.8

(22) Date of Filing 28.03.1998

(51) INT CL[6]
G06F 12/14 9/455 11/00

(52) UK CL (Edition Q)
G4A AAP AFMP

(56) Documents Cited
WO 95/26000 A1 US 5675645 A US 4262329 A

(58) Field of Search
UK CL (Edition P) G4A AAP AFMP APL
INT CL[6] G06F 1/00 9/455 11/00 12/14
Online: WPI, INSPEC, COMPUTER

(71) Applicant(s)
Motorola Limited
(Incorporated in the United Kingdom)
Jays Close, Viables Industrial Estate, BASINGSTOKE, Hampshire, RG22 4PD, United Kingdom

(72) Inventor(s)
Stephen McPhillie
Matthew Dickie
Albert Dye

(74) Agent and/or Address for Service
Sarah Gibson
Motorola Limited, European Intellectual Property Operation, Midpoint, Alencon Link, BASINGSTOKE, Hampshire, RG21 7PL, United Kingdom

(54) Abstract Title
Maintaining security in development tools

(57) An authentication process within a development tool such as an emulator 30 or a simulator, enables the control of access to the cryptographic functions of the semiconductor target device, for example a smart card microcontroller (100, fig. 4). Emulator 30 includes a secure processor 46 and an unsecure processor 44 to emulate the target IC. The cryptographic functionality of the target device resides with the secure processor 44, whose memory configuration (fig. 5) is similar to that of the target device. The developer can only access the secure processor via the unsecure processor, and an authentication process (fig. 6) is used to establish communications between the two processors within the emulator whenever the user's application, executed within the unsecure processor 44, requires execution of a cryptography function. Whereas in a traditional "open" emulator environment, using a device in "expanded mode", the developer has access to explore and alter any cryptographic restrictions, here this access is restricted to comply with existing and possibly forthcoming government agency restrictions.

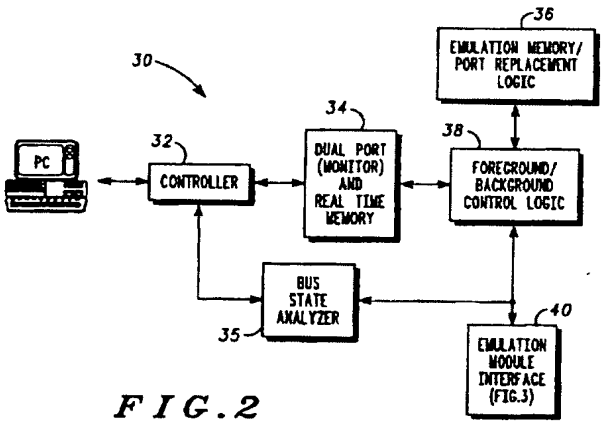

FIG.2

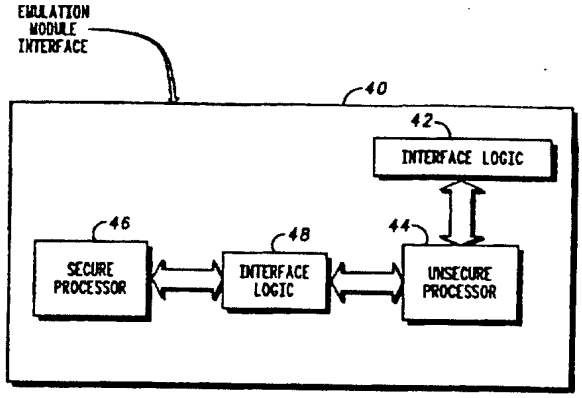

FIG.3

GB 2 336 005 A

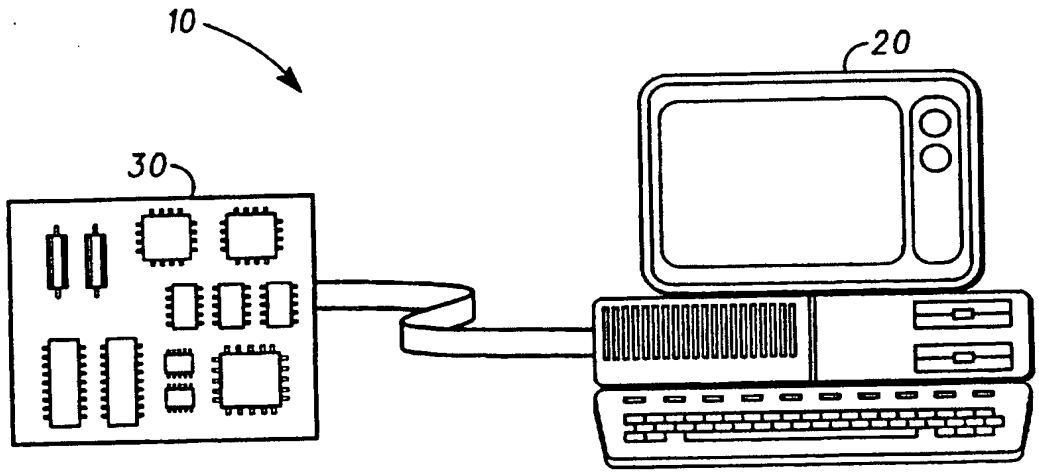
10
20
30

FIG.1

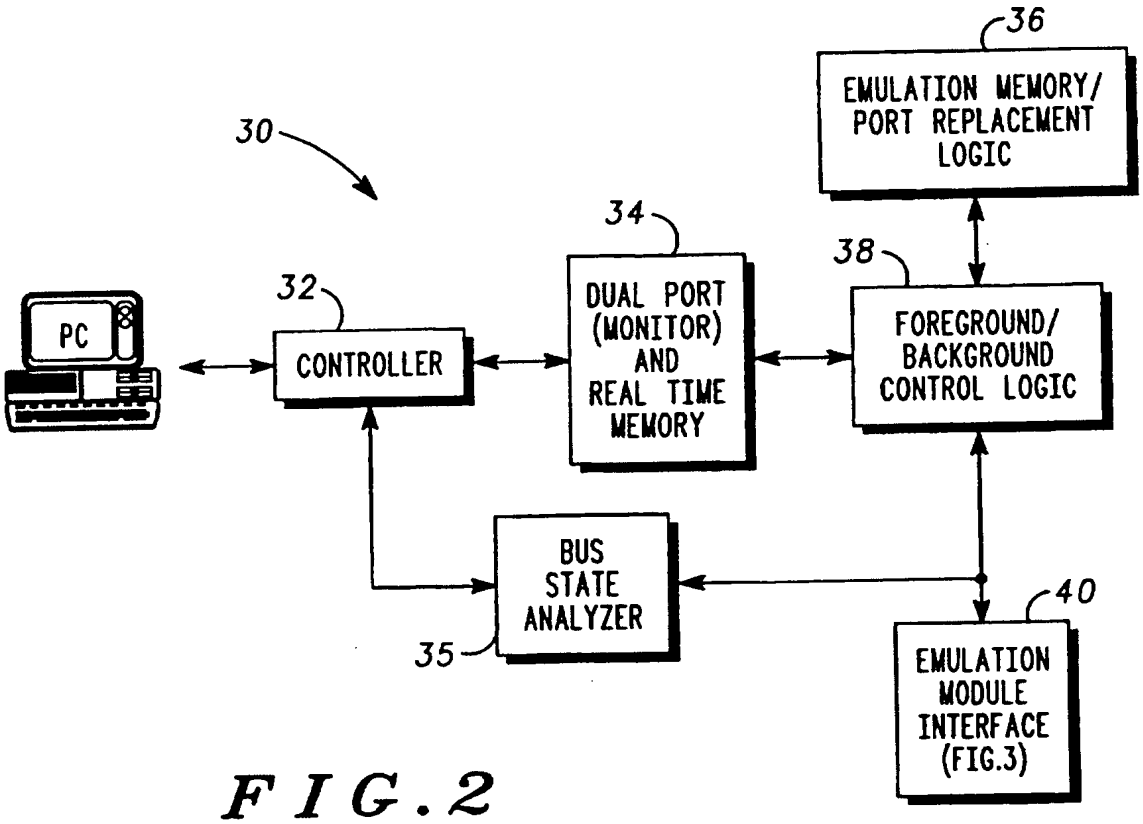
36
EMULATION MEMORY/ PORT REPLACEMENT LOGIC
30
34
38
32
PC
CONTROLLER
DUAL PORT (MONITOR) AND REAL TIME MEMORY
FOREGROUND/ BACKGROUND CONTROL LOGIC
BUS STATE ANALYZER
35
40
EMULATION MODULE INTERFACE (FIG.3)

FIG.2

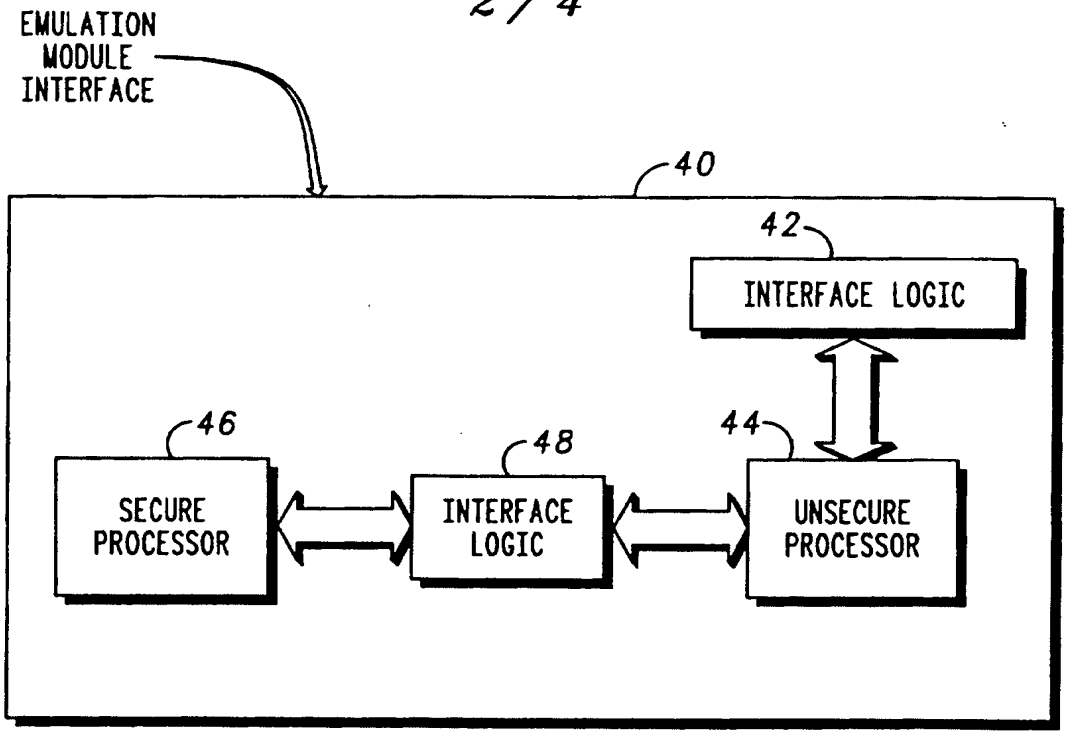

FIG. 3

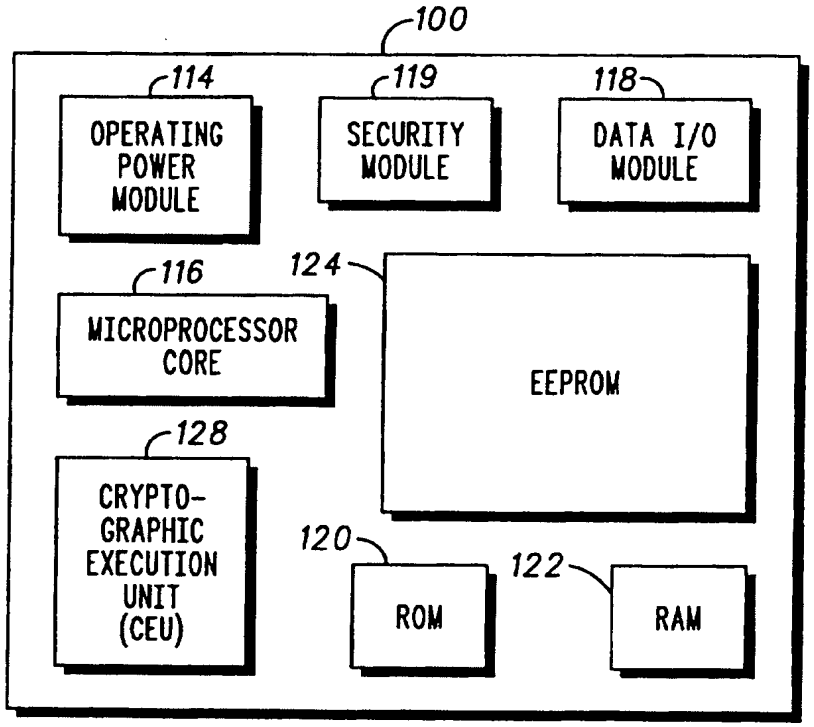

FIG. 4

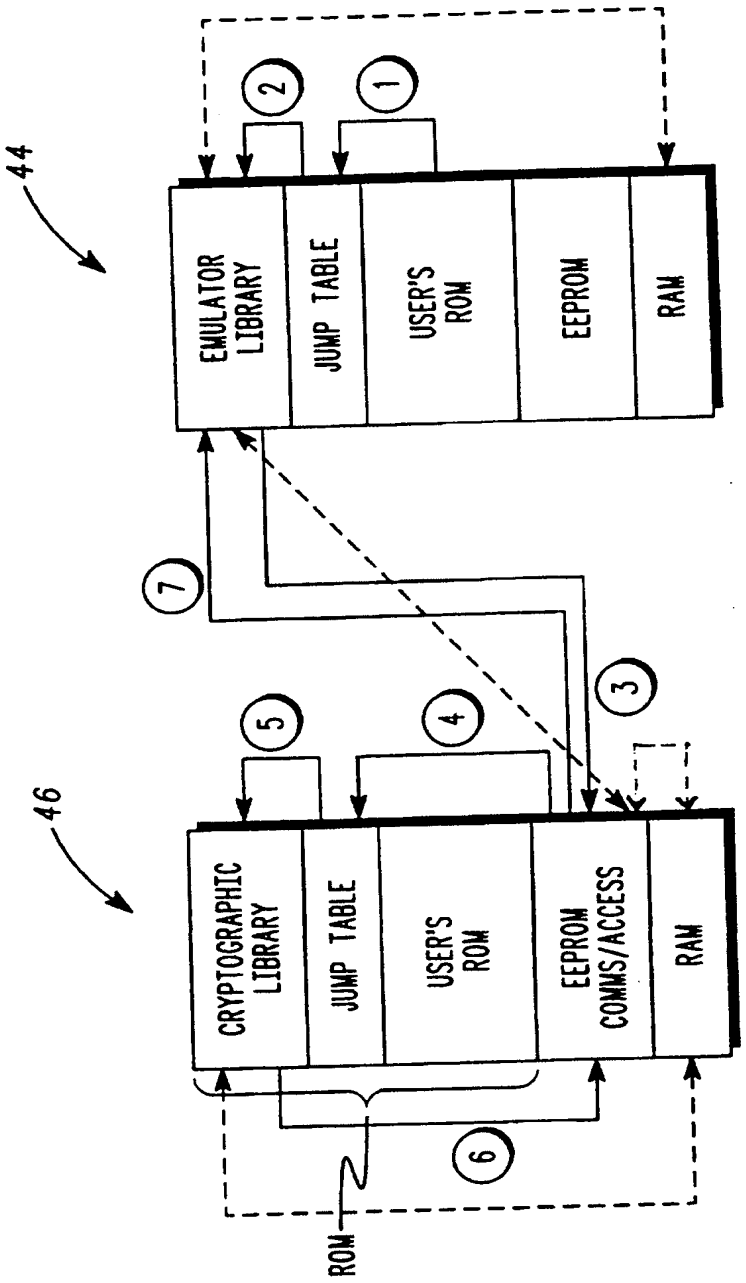
44
EMULATOR LIBRARY
JUMP TABLE
USER'S ROM
EEPROM
RAM
2
1
7
3
5
4
46
CRYPTOGRAPHIC LIBRARY
JUMP TABLE
USER'S ROM
EEPROM COMMS/ACCESS
RAM
6
ROM

FIG.5

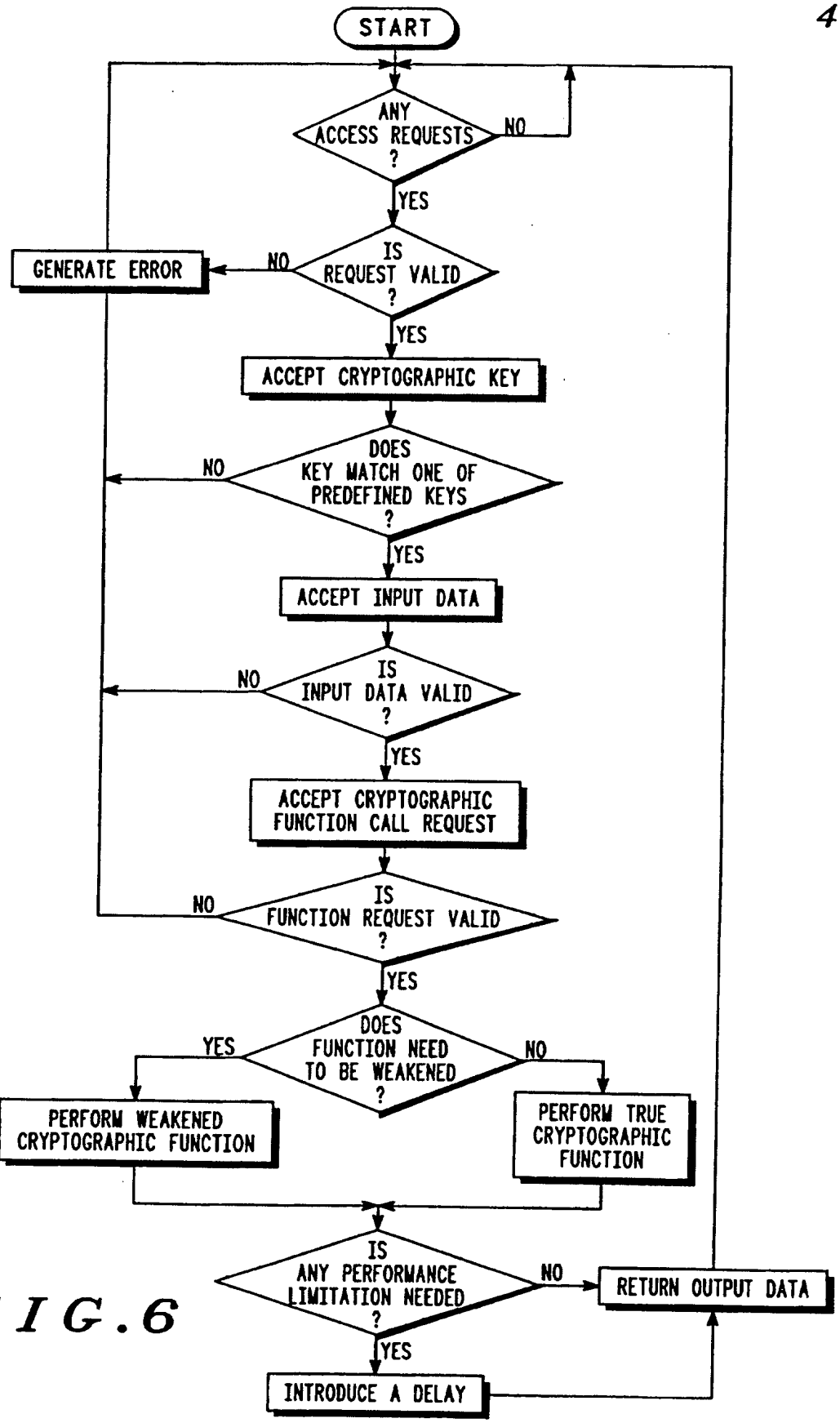

FIG.6

METHOD FOR MAINTAINING SECURITY IN A DATA PROCESSING SYSTEM AND DATA PROCESSING SYSTEM THEREFOR

Field of the Invention

The present invention relates generally to maintaining security in data processing systems, and more particularly to maintaining security in development tools used in conjunction with encryption devices.

Background of the Invention

Cryptographic techniques are increasingly being used to ensure information security during various data transfers, and such techniques are becoming increasingly "strong" as technology advances. A common example is the use of cryptography is smart card applications. A smart card might contain sensitive financial or health information about the card holder, or information which is highly valuable to the card issuer, such that the ability to protect against an unwanted interception of the information is of critical importance.

Cryptographic technology is advancing quickly, so much so that national governments are becoming more and more concerned about potential abuses of such techniques (i.e. using the techniques for unlawful purposes). In some instances, governments are imposing particular restrictions on cryptographic capabilities of products. Smart card manufacturers, and more particularly semiconductor manufacturers who supply the card manufactures with encryption devices for the cards, are therefore having to closely monitor their activity with a variety of government agencies in an attempt to reach a commercially viable solution to the problem. The activity affected is not only tied to the cryptographic capabilities of the semiconductor devices, but also to complementary products supplied by the semiconductor manufacture in the form of development tools.

Generally, a development tool is a hardware and/or software tool used to develop computer programs and applications for use in conjunction with a particular semiconductor device. Examples of software development tools include text editors, assemblers, debug monitors and simulators. Hardware development tools include emulators, logic analysers, and programmable memory programmers. Of particular concern to government agencies from a security perspective are emulators of smart card devices.

An emulator is a real-time development tool built around the actual microcontroller or microprocessor used as the smart card device. An emulator can execute program instructions exactly as they would be executed in the finished application. A typical emulator board includes the actual device, albeit in an "expanded mode," whereby the user has the ability to "get inside" the device and to explore and even change the functionality of the device being emulated. Emulators are used by semiconductor customers or third party software developers to develop application programs for the semiconductor device. Customer's also use an emulator to develop their own customer-specific code for the finished device, which the semiconductor manufacturer can then incorporate into the memory of the actual device during manufacturing.

The availability of emulators play a critical role in a customer's ability and desire to use a particular microcontroller or microprocessor in their application. Thus, it is in a semiconductor manufacturer's interest to make suitable emulators available, and in a timely manner. However, existing and contemplated government agency restrictions on the "openness" of emulators for smart card devices threaten a manufacturer's ability to provide such tools.

A few options exist that address the government agencies' concern relative to smart card emulation tools. A first option is to slow the cryptographic functionality down to a rate which is acceptable to the government. Generally, the government is less concerned when encryption/decryption takes longer (e.g. less than 2 kilo-bits per second). A second option is to weaken the algorithm of the emulator as compared to the actual device. The strength of the encryption is measured by the

number of bits of the encryption/decryption key. The shorter the key, the less likely the occurrence of an unlawful use because of the relative ease of cracking it. A third option is for the semiconductor manufacturer to design the emulation tool such that it uses only a limited number of fixed keys or predictable keys. The fixed/predictable key option enables the developer to use the true algorithms in real-time, and is likely to satisfy government agencies because cracking encryptions would be relatively easy in view of the fact that only a small number of keys can be used. However, in a conventional development system which is intentionally designed to be open to the developer, any of the above-mentioned options can be circumvented by the developer. The open access to the actual device within a conventional emulator, for example, would permit the developer to change the rate of encryption, the strength of encryption, and add to the predefined number of permissible keys, rendering any restrictions formerly designed into the device ineffective.

In view of the foregoing, it is apparent that a need exists to satisfy government agencies that the security of development tools for encryption devices, and more specifically of smart card devices, can be maintained while at the same time keeping in mind the need of developers to precisely emulate or simulate the final smart card device.

Brief Description of the Drawings

FIG. 1 generally illustrates a development system, such as an emulator, as would be used by a semiconductor manufacturer's customer or third party software developer interested in writing code that runs on one of the manufacturer's microcontrollers or microprocessors.

FIG. 2 is a functional block diagram of the development system of FIG 1.

FIG. 3 is a more-detailed view of the emulation module interface (EMI) block of FIG. 2, also in block diagram form, and in accordance with one embodiment of the present invention.

FIG. 4 is a functional block diagram illustration of an integrated circuit as might be used in a smart card.

FIG. 5 is a block diagram of a portion of the secure and the unsecure processors of the EMI of FIG. 3 which illustrates the manner in which the two processors communicate with one another to perform cryptographic functions in accordance with one specific embodiment of the present invention.

FIG. 6 is a flow diagram representing the manner in which access to the secure processor of FIGs. 3 and 5 is controlled in accordance with one specific embodiment of the invention.

These and other features, and advantages, will be more clearly understood from the following detailed description taken in conjunction with the accompanying drawings. It is important to point out that the illustrations may not necessarily be drawn to scale, and that there may be other embodiments of the present invention which are not specifically illustrated. Where appropriate, like reference numerals may be used to designate identical or corresponding parts throughout the several views.

Detailed Description of a Preferred Embodiment

Generally, the present invention enables access control of cryptographic functions of a semiconductor device to be imposed by an authentication process within a development tool. The authentication process satisfies government agency concerns regarding possible abuses of cryptography by the developer, while at the same time permits the developer to fully emulate or simulate the target device. This is achieved through the use of two processors within the development tool, one of which is a rendition of the processor to be emulated or simulated without direct access to the cryptographic functions, and the second is a physical (i.e. hardware, such as an integrated circuit) secure processor which includes the cryptographic functions and libraries of the processor being emulated. The developer has direct access and control over the first processor, the "unsecure" processor, but if a routine calls for execution of a cryptographic function, the unsecure processor makes a request to the secure processor to execute the function. After authenticating the requested function, the secure processor performs the function, usually using data transmitted to it by the unsecure processor, transmits the results to the unsecure processor, and returns control thereto. Security is maintained because the developer has no direct access to the protected portion of the secure processor.

FIGs. 1-6 will be used to describe various attributes of the invention in greater detail. It is noted that while a specific implementation of the invention is described, the invention is not intended to be limited to the described implementation. For example, the embodiment described uses the invention in conjunction with a development system. However, the present invention can be used in conjunction in any data processing system, either in a real application environment or in a development environment, wherein access to particular functionality or data must be restricted and maintained. Also, the development system herein described is an emulator, although the invention's principles can be extended to other development tools, for instance simulators. Therefore, it is intended that the invention encompass such variations and modifications, as well as others described herein, and those which fall within the scope of the appended claims

FIG. 1 is a general illustration of a development system 10. System 10 includes a personal computer (PC) 20 connected to a development board, which in this example is an emulator 30. A customer or third party software developer (hereinafter collectively called a user or developer) instructs and monitors the operation of emulator 30 using the PC as the interface.

FIG. 2 illustrates a functional block-diagram of emulator 30. Emulator 30 includes a controller 32 which is coupled to dual-port/real-time memory 34 and a bus state analyser 35. The controller is responsible for communication with the host PC and the target device. As used herein, target device refers to the particular semiconductor device which the emulator is emulating. In the particular example explained in more detail below, the target device is a smart card microcontroller with cryptographic functionality such as that shown in FIG. 4. Real-time memory 34 provides a common memory area which can be accessed by both the controller 32 and the target device, and bus state analyser 35 captures and stores address, data, and control signal information into a buffer which the developer can display on the PC.

Emulator 30 also includes a block 36 having emulation memory and port replacement logic. Emulation memory provides a pseudo read-only memory (ROM) area in which the user can develop and store software for the target device, while port replacement logic is included to rebuild the inputs/outputs (I/O) of the target device which are lost when the device is configured in test mode (e.g. during emulation). Emulator 30 also includes foreground/background control logic 38 coupled to block 36 and to dual-port/real-time memory 34. Control logic 38 controls whether the target device is in foreground or background mode. In foreground mode, the target device executes user code residing in emulation memory 36. In background mode, the target device executes user code residing in dual-port/real-time memory 34. Finally, emulator 30 includes an emulation module interface (EMI) 40 which is coupled to both the bus state analyser 35 and to control logic 38. For more information regarding the general construction and operation of an emulator, the reader is directed to the Motorola Modular Development System Operations Manual (Document

No. MMDS05OM/D, Motorola, Inc. 1993), which describes an emulator for Motorola's microcontroller products, and which is hereby incorporated by reference. An emulator in accordance with the present invention operates in much the same manner, with a principal exception being the EMI block and the way in which access to the target device is restricted.

EMI 40 is illustrated in more detail in FIG. 3. In accordance with an embodiment of the present invention, EMI 40 includes two distinct processors. As used herein a processor refers to either a microcontroller or microprocessor implemented as an integrated circuit, when used in a physical sense. As noted later, a processor can also be a software simulation of the a physical processor. A first processor is referred to as an unsecure processor 44, and the second is referred to as a secure processor 46. A "secure" processor can generally be thought of as a processor which resists physical and logical attacks (of a one-off or systematic nature) on the data contained within the processor. The minimum time and cost to an attacker to infiltrate the protected data is sufficiently high to render these attacks unviable in benefit of any kind to the attacker or other beneficiary. In theory, given enough time and resources, a hacker can infiltrate the data. No system is one hundred percent secure. Therefore the terms "secure" and "unsecure" are used in a more practical and relative sense, but there is a general understanding within the industry of what a "secure" (and sometime referred to as a "security") processor is. An "unsecure" processor is simply one which is not secure.

Within EMI 40, interface logic 42 provides the interface necessary to permit communication between the unsecure processor 44 and the rest of emulator 30, while interface logic 48 provides the interface necessary to permit communication between the two processors. Note that that interface logic 42 is not directly linked to the secure processor. When the invention is used in a development system environment, the developer only has access to the secure processor via the unsecure processor. Accordingly, restrictions are able to be imposed on the cryptography functions (residing within the secure processor) without the developer being able to alter such restrictions. The "open" environment of the emulator is thus constrained to the unsecure processor. However, the user is still able to emulate cryptography functions using the secure

processor as described subsequently. Communications between the two processors are controlled by an authentication process, also described below.

FIG. 4 illustrates, at a block level, the basic constituency of an integrated circuit (IC) 100 as might be used in a smart card. Thus, for purposes of illustrating an example of the present invention, IC 100 would be a target device. In a smart card, IC 100 would be used to transmit encrypted data to a reader terminal and to decrypt data received from the terminal. Much of the operation of IC 100 is not important for purposes of understanding the invention, however a brief explanation of the primary functions of the IC may be useful.

An operating power module 114 receives operating power by either direct contact with the reader terminal via contact pads on the card (called contact mode), or by radio frequency (RF) transmission via an antennae. Operating power module 114 then distributes a positive power supply potential (e.g. V_{DD}) to the other circuitry in the IC. A processor core 116 performs the control, timing, and decision making functions of the card. For example, processor 116 controls the read, write and erase operations to the memory and makes data available to data I/O module 118. Data I/O module 118 sends and receives data to and from the reader. Security module 119 prevents unauthorised use or access of the card and routinely checks operating integrity.

Read-Only-Memory (ROM) module 120 of IC 100 stores the program instructions for the given application which are set during the manufacturing process and are executed by processor core 116. As mentioned above, the ROM code is usually customer or application specific, thus the need for an emulation tool to facilitate proper coding. A Random-Access-Memory (RAM) module 122 is also included. RAM is volatile memory and thus provides temporary storage of information. Electrically Erasable Programmable ROM (EEPROM) 124 is a non-volatile memory array of IC 100 that stores the primary information of the card, such as personal identification, medical history, banking information, monetary values, security codes, etc. depending on the application. While

EEPROM is a preferred form of memory, other types of non-volatile memory can be used instead.

IC 100 also includes a Cryptographic Execution Unit (CEU) 128 which is used to encrypt data being transferred from the card to a reader and to decrypt data being transferred from the reader to the card. The manner in which data is encrypted or decrypted is not particularly important for the purposes of understanding and practising the present invention. Conventional cryptographic methods can be employed (e.g. RSA cryptography). Accordingly, a detailed explanation of such methods is omitted.

While IC 100 has been illustrated to include definitive blocks for performing particular functions, it is noted that in practice the particular functional blocks of even a smart card IC may in fact not be clearly identifiable as "blocks" on the actual manufactured IC. Furthermore the arrangement of such "blocks" on the chip are likely to not correspond to the arrangement shown. Furthermore, an IC in a card may include functional blocks other than those illustrated in FIG. 4, such as a charge pump. Accordingly, the figures are not intended to limit the scope of the invention unless expressly indicated otherwise.

In accordance with one implementation of the present invention, IC 100 of FIG. 4 is emulated by emulator 30 having EMI 40 as described above in reference to FIGs. 2 and 3. In this implementation, secure processor 46 of FIG. 3 would functionally be quite similar to IC 100. Unsecure processor 44 of FIG. 3 would be similar, but would lack the ability to perform cryptographic functions by itself (i.e. the CEU 128 would not be included in the unsecure processor). Security module 119 of IC 100 monitors things such as operating temperature and frequency to ensure proper use of the card. While security module 119 functions to prevents unauthorised access of the card, it does not perform cryptography and thus can be included on the unsecure processor. While both the secure and unsecure processors will contain RAM, ROM, and in this example EEPROM, the contents of the memory arrays will differ. For example, the memory of the secure processor will contain the cryptography routines,

but the memory of the unsecure processor will not. This is illustrated in more detail in FIG. 5.

FIG. 5 illustrates the relevant memory portions of secure processor 46 and unsecure processor 44. It is noted that while the illustration and description of FIG. 5 refer to information being stored particularly in either ROM, RAM or EEPROM of the processors, the particular type of memory which is used to store the information is not restricted by the invention. For example that which is described as being stored in EEPROM can instead be stored in ROM. Thus, FIG. 5 and its description should be viewed as an example of a particular implementation of the invention provided by way of example.

As shown in FIG. 5, the ROM of secure processor 46 is segmented into three distinct portions: 1) a portion dedicated for the user's code; 2) the cryptography library which contains the cryptography routines; and 3) a jump table. Not all smart card processors will utilise a jump table to perform cryptography, but such a table is used to execute cryptography functions in Motorola's smart card devices and thus is used in this example. Furthermore, there may be dedicated or reserved portions of the ROM set aside for other purposes, but the three portions identified in FIG. 5 are sufficient for understanding the operation of the present invention. Secure processor 46 also includes RAM and a non-volatile memory portion in the form of EEPROM. In the secure processor of this example, the EEPROM is used to store the software program which performs the authentication process, which is represented in FIG. 5 as "Comms/Access" standing for Communication and Access. The authentication process controls access to the secure processor and communication between the two processors as explained further below.

The memory configuration of secure processor 46 is similar to that of the actual target device being emulated. The ROM of the target device will have a large portion devoted to the user's code, but will have a portion of it reserved by the semiconductor manufacturer. The manufacturer's reserved portion will contain the cryptography routines. Again, in Motorola's particular implementation the calls to the cryptography library are made via a jump table, also stored in ROM. A

difference between the memory blocks of secure processor 46 and the target device is that the memory of the target device will not have a need for the Communication and Access logic (i.e. the authentication process) since the target device is not operated in an "open" emulation environment in which one can extract and modify the memory contents.

Unsecure processor 44 likewise includes RAM and EEPROM, but in this instance the memory is not programmed to include the Communication and Access control software. As FIG. 5 also illustrates, the ROM contents of the unsecure processor in this particular example (which in an emulation environment resides in the emulation memory 36) also is segmented into three portions: 1) the user's ROM; 2) a jump table; and 3) an emulator library. The emulator library serves as the communication driver to secure processor 46, passing data and instructions thereto. Note that the ROM of unsecure processor 44 does not contain a cryptographic library. Thus, the developer does not have direct access to the contents of such library.

The communication and data flow between the two processor is now described with reference to FIG. 5. A reader can follow the paths generally using the sequentially numbered arrows. In the emulator, the user's application program will be executed from ROM or other memory within the unsecure processor 44. When the routine requires execution of a cryptography function, the program is directed to a location in the jump table of the unsecure processor (Path 1), pursuant to the programmer's instruction manual for the particular device being emulated. In the target device, the jump table would redirect the call to a particular location of the cryptography library. But, in accordance with the present invention, the developer is prevented from having direct access to the cryptography library. Instead, the jump table directs the call to the emulator library (Path 2). It is noted that the emulator library should be organised as if it were the actual cryptography library so that the same jump table can be used in the emulation environment as in the actual operation environment. As such, from a programmer's model, the cryptography function calls can be identical between the emulator environment and the real-world device environment.

In response to the jump to the emulator library, unsecure processor 44 then attempts to communicate with secure processor 46 to request execution of the called cryptographic function. As stated above, all such communications are controlled by the Communication and Access program (Path 3). If communication is to be permitted (as explained further in reference to FIG. 6), the authentication routine then calls a jump table location (Path 4) corresponding to the original jump table call by the user. The jump table call in the secure processor invokes execution of a particular cryptography function stored in ROM, or other memory, of the secure device (Path 5). After execution of this function, the result is feed back to the EEPROM of the secure processor (Path 6) because all information transferred to the unsecure processor (Path 7) is controlled by the Communication and Access program (which by the way can be stored in a type of memory other than EEPROM) . It is noted that during various steps in the above-described program flow, the RAM contents of one or both of the processors may be accessed and transferred. Accordingly, the RAM portions of both processors show dashed line data transfer links. For example, an encryption process performed by secure processor 46 will need the data which is to be encrypted. This data will likely be stored initially in RAM of the unsecure processor, and then transferred to RAM of the secure processor, again via the Comunication and Access logic.

FIG. 6 is a flow diagram representing an example authentication sequence used by the Communication and Access logic of secure processor 46 to establish communication between the two processors within the emulator. It is noted that the particular sequence shown includes four different options of controlling access to and operation of the cryptography function. These correspond to four ways in which the cryptographic capability of the device can be controlled to satisfy restrictions imposed by government agencies (namely: use of fixed or predictable keys; use of a "weak" algorithm; use of "slow" encryption/decryption; and limiting use to particular types of cryptography functions). However, there are likely to be other methods of complying with governmental restrictions which likewise need to be safeguarded from alteration in a development tool environment. Thus, one should recognise that the flow in FIG. 6 is but one example of how security of the device can be maintained. Any one of these restrictions may satisfy a particular government agency's

requirement, or a restriction other than the four discussed may be imposed. For example, use of a fixed or predictable key alone may be sufficient. Having the security control residing in a physically separate and secure processor which cannot be directly accessed by the developer ensures that the development tool cannot be altered to operate beyond the applicable restrictions imposed by government agencies, regardless of manner in which the semiconductor manufacturer chooses to abide by such restrictions.

Continuing with the example of FIG. 6, whenever an access of the secure processor is attempted, the authentication process should first determine if the request is a recognised request. If not, the request is rejected and an error message is generated. If the request is accepted, the authentication process will then wait for a cryptographic key to be received from the unsecure processor. The cryptographic key should be incorporated into a valid message format, as an invalid format will cause the entire access request to be rejected. Once received, the cryptographic key is compared against a predefined set of keys held in the secure processor's memory. This is the first option of security restrictions imposed upon the cryptography process. If the key does not match any of the predefined keys, the key and therefore the whole request for access sequence is rejected and an error message is generated. If the key is accepted, the authentication process will then wait for input data to be received from the unsecure processor. Again, input data should be incorporated into a valid message form, as an invalid form will cause the entire access request to be rejected. Once received, the input data will be placed into authorised areas of the processor's memory. Any attempt to place data in an unauthorised area of memory will cause the data, and thus the whole request for access to be rejected.

If the input data is accepted, the authentication process will then wait for an execute command to be received from the unsecure processor. Execute commands should likewise be checked for a valid message format, and an error message generated if formatting is improper. The second option of security restriction is implemented by programming, at the time of processor manufacture, a list of permitted cryptographic functions into the secure processor's memory. This list cannot be altered at a later date,

but can be tailored at the manufacturing stage to meet particular customer and government agency requirements. The authentication process will use this list to determine if the cryptographic function for which an execution request has been received can be executed. If permissible, the secure processor will then perform the appropriate cryptographic function. If impermissible, the whole access request sequence is rejected and an error message is generated.

Before performing the cryptographic function, a third option of security restriction can be introduced, namely a "weakened" algorithm can be used in place of the true algorithm which would be used in the target device in a real-world application. Whether a weakened algorithm should be used might depend on the particular function requested to be performed, or might apply to all functions. Again, the degree of weakening might depend on the government agency controlling the use of the system, and thus can be tailored ahead of time by the manufacturer but cannot be altered by the developer.

A fourth option of security can be imposed after the cryptographic function is performed but prior to transmitting the information back to the unsecure processor. This can be accomplished by introducing a delay into the communication. For example, the cryptographic execution unit of the secure processor may be able to encrypt data at a rate of 5 kilobits per second (Kb/s), but the relevant government may impose a restriction of 2Kb/s. In such a case, the Communication and Access program of the secure processor can introduce an intentional delay to met the restriction, yet the developer will not have access to this code to circumvent the restriction.

After successfully progressing through the various stages of the authentication process, the encrypted/decrypted data is transferred from the EEPROM (Communication and Access logic) of the secure processor to the emulator library of the unsecure processor. The developer's routine stored in the user ROM, or other memory, of the unsecure processor continues, using the resulting encrypted/decrypted data as needed. The process of FIG. 6 then repeats each time access to the secure processor is requested within the routine.

From the foregoing description and illustrations contained herein it is apparent that the present invention provides a way to satisfy government agencies with respect to possible abuses of cryptography techniques in development tools. The present invention imposes appropriate restrictions on the cryptographic capability of a tool, and includes the cryptography functions of the tool in a physical, secure processor which cannot be tampered with by the developer. Accordingly, the need for getting government approval for export of such a tool, which can be time-consuming and costly, can be avoided, providing a tremendous time-to-market advantage for the tool provider. A particularly advantageous implementation of the present invention uses only the fixed key or predictable key option of security in a development tool, thereby permitting the developer to emulate the target device with real-strength and real-time encryption algorithms.

As noted earlier, the invention has been described and illustrated with reference to specific embodiments thereof, however it is not intended that the invention be limited to these illustrative embodiments. Those skilled in the art will recognise that modifications and variations can be made without departing from the invention. For example, the present invention is not limited to an emulator environment but rather is applicable in any data processing system wherein a user/programmer is only selectively permitted to access portions of a physical, secure processor. Also, if used in an emulator, the invention may not necessarily be implemented within the EMI. Other manufacturer's development tools may be designed differently and require modification accordingly. Furthermore, the invention is not limited to use with a processor having cryptographic capability. Simple data in a ROM may benefit from a similar type of access protection. It is also noted that the invention need not be practiced using a physical unsecure processor. Firstly, the "unsecure processor" may be a software simulation of a physical unsecure processor. In this case, the development tool is a simulator rather than an emulator, with the "unsecure processor" being a software version of the processor and the secure processor continuing to be a physical (e.g. IC) processor. Furthermore, the "unsecure processor" could in fact be a programmable device such as a Field Programmable Gate Array (FPGA) or other type of

gate array which performs the traditional processor functions. Moreover, the "unsecure processor" need not be "unsecure" but simply have a level of security which is lower than that of the "secure processor". In other words, two secure processors can be used in a data processing system as described, but one processor has a higher level of security associated with it (and more particularly with the data to be protected) than the other processor. Therefore, it is intended that this invention encompass all such variations and modifications as fall within the scope of the appended claims.

Claims

1. A method for maintaining security in a data processing system comprising the steps of:

providing a data processing system comprising a first data processor and a second data processor, wherein the second data processor has a portion to be protected against user access and has a higher security level than the first data processor, and wherein the first data processor is operably coupled to the second data processor to permit selective communication therebetween;

executing a routine with the first data processor which requires access to the portion of the second data processor;

transmitting a request for access from the first data processor to the second data processor;

using the second data processor to access the portion of the second data processor in response to the request when the request is valid; and

transmitting a reply to the request from the second data processor to the first data processor.

2. A data processing system comprising:

a first data processor and a second data processor, wherein the second data processor has a portion to be protected against user access and has a security level higher than the first data processor, and wherein the first data processor is operably coupled to the second data processor to permit selective communication therebetween.

3. The method of claim 1 or the system of claim 2 wherein the first data processor is a unsecure data processor and the second data processor is a secure data processor.

4. The method or system 3 wherein the data processing system is a development system.

5. The method or system of claim 4 wherein the development system is an emulator.

6. The method or system of claim 4 wherein the first data processor is a software simulation of a physical data processor and wherein the development system is a simulator.

7. The method or system of claim 4 wherein the development system is a development system for a semiconductor device capable of executing cryptography functions.

8. The method or system of claim 4 wherein the development system is a development system for a smart card device.

9. The method or system of claim 1, 2, or 4 wherein the second data processor is capable of performing a cryptographic function, and wherein performance of the cryptographic function comprises execution of a routine stored in memory of the second data processor.

10. The method or system of claim 9 wherein the routine is called by a jump table also stored in memory of the second date processor.

11. The method or system of claim 1 or 2 wherein the first data processor is a secure data processor and wherein the second data processor is more secure with respect to the portion to be protected against user access than the first data processor.

12. A method for maintaining security in a data processing system as hereinbefore described with reference to the accompanying figures.

13. A data processing system as hereinbefore described with reference to the accompanying figures.

The Patent Office

19

Application No: GB 9806575.8
Claims searched: 1-13

Examiner: Melanie Gee
Date of search: 22 September 1998

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.P): G4A (AAP, AFMP, APL)

Int Cl (Ed.6): G06F 1/00, 9/455, 11/00, 12/14

Other: Online: WPI, INSPEC, COMPUTER

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
A	WO 95/26000 A1	(VIEWLOGIC SYSTEMS), see whole document.	
X	US 5675645 A	(SHWARTZ et al.), see fig. 1 and col. 9 line 58 - col. 11 line 3.	1 & 2
X	US 4262329 A	(BRIGHT et al.), see whole document.	1-3, 9

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.